

## Multi-Deck Input Capability Upgraded

George Mesina

September, 2015

*Improvements made to the code's multi-deck input capability correct errors and increase its applicability.*

### Background

A little known feature of RELAP5-3D is the capability to incorporate more than one input deck in a single input file. This feature is used in three of the standard input tests included in the installation problem set. One of them, 3DFLOW, runs 18 input cases by employing two separate input decks that have 9 cases each.

The "." terminator card specifies the end of an input deck and typically marks the end of an input file as well. Some users have noticed that if a blank line follows this card, the code runs the input model to completion, but then prints out an error message. The reason this happens is that the code treats lines in the file that follow the terminator card as if they belong to a subsequent input deck.

### Uses for Multi-Deck Input

Multi-deck input files can be used for many purposes.

- Running a large problem with a first deck followed by a second deck that performs a strip can produce a plot file small enough to email yet still contain the relevant data.
- In older versions of RELAP5, a second run could exercise the code's internal plot feature to produce graphs of key variables; this feature is not present in recent versions of RELAP5-3D because so many superior plot programs are available.
- One can load balance a collection of input decks into multi-deck files so that RELAP5-3D runs each multi-deck file in about the same amount of time on a multi-core computer.
- Multi-deck is useful for parameter studies. Typically, parameter studies are carried out with multi-case input decks; however, sometimes this is insufficient. The user request in the next section is one example.

### Sequential Parameter Study

RELAP5-3D users wanted to perform a parameter study of RELAP5-3D coupled to another program running a sequence of input decks where each one restarted from the end of the previous problem in

the sequence. The sequence of runs represented a run to steady state, a renodalization, a change of the input of the coupled program, and finally a transient calculation. The parameter study involved changing some parameters in each of the sequential decks. It required such a large number of variations that it merited use of a cluster supercomputer.

Finally for each choice of parameters, the users required that the sequence of input decks had to all run on a single process thread. This requirement meant that RELAP5-3D could not run the first deck to completion, then make a new run to do the renodalization restart, etc. The code had to keep running without stopping until all decks in the sequence pertaining to the parameter selection were run. RELAP5-3D has two means to effectively run multiple decks without terminating its run, multi-case and mutli-deck input files.

Multi-case input cannot solve this problem. In a multi-case run, each case runs to completion, but then memory is cleansed and, after processing the input for the next case, the transient calculation begins from initial conditions, not the final conditions of the previous run.

Thus multi-deck input had to be used. Each deck in the sequence had to restart from the end of the previous calculation. When this was first attempted, some errors were discovered with restarting a sequence of input in multi-deck files. These errors have been fixed.

## **Error Resolution**

The first error involved the plot file. The error message indicated that the code recognized the plot file format correctly early in input processing, but then could not read the file. This was traced to the file not being rewound either at the end of the first calculation, or when the existing plot file was about to be opened. This is not necessary when RELAP5-3D runs to completion, then restarts from a separate run. The operating system defaults to opening files from their beginnings when a new process seeks to open them.

Older versions of RELAP5 were examined. They rewound the plot file at the end of the transient when closing it. This solution was implemented and solved the problem.

The second error occurred when the third deck in sequence tried to use the restart file from the second deck, which itself had restarted the first deck. Failures traced back to different input processing subroutines depending on the base model involved. The problem was traced to the way the second run wrote the restart file.

Restart runs search restart files for the record requested by the user, either by integer advancement number of floating point cumulative time. When the record is found, the search terminates with all input from that time in the transient having been read into the proper memory locations. Thereafter input processing may add or delete components or tables, renodalize, or make other changes. This necessitates writing the data on the restart file immediately after input processing because it has become the initial data of the new configuration of the input model.

Unfortunately, this means that there exist two records at the restart time, the final one for the initial configuration of the model, and the new one that serves as the initial configuration of the modified model. When the code tried to read past the first record, it had trouble. It should be noted that when the code started running fresh and opened a file with two such records, it had no problems. This indicated an unresolved pointer or memory deallocation issue, an issue that could be time-consuming to find. Unfortunately, the users who requested the sequential restart capability had a project with serious time constraints.

A quick solution was to eliminate the first of the two records at the restart time. The algorithmic change is modest. Simply return to the start of the records associated with the restart time before writing new data on the restart file.

The coding featured multiple go-to statements including obsolescent arithmetic go-to structures. This was rewritten with modern constructs and the internal documentation was improved. Once the algorithmic modification was made, the code ran correctly for the users' needs.

Two new test cases have been added to the installation test set to exercise the multi-deck capability. It is noted that all restart records from before the restart time are preserved and may be accessed through either advancement number or cumulative time.

## **Summary**

The code capability to run multiple input decks in a single file, without have to terminate one run before running the next, has been refurbished and tested. This has created a new capability to run a sequence of restart calculations from a single file.